Announcements
▶ Discussion Friday
▶ No class monday (President’s day)
▶ Akshay’s Office hours Tu 5-6 just next week
▶ Midterm two weeks from today (I will post a practice exam)

Recap
▶ Greedy algorithms for interval scheduling
▶ Interval scheduling with no conflicts
▶ Interval scheduling minimizing number of rooms
▶ Minimizing maximum lateness
▶ Observation: problems have different combinatorial structure.

Shortest Paths Problem
▶ Given a weighted directed graph, let $w(e) > 0$ denote the length of edge $e$ and for a path $P$ consisting of edges $e_1, e_2, \ldots, e_k$ we denote the length of this path as $\ell(P) = w(e_1) + w(e_2) + \ldots + w(e_k)$
▶ Fix a node $s$ and let $d(v)$ be the length of shortest $s \rightarrow v$ path.
▶ Problem: Can we efficiently find $d(v)$ for all nodes $v \in V$?

A special case
▶ Question: What if all edges have weight $w(e) = 1$?
▶ Answer: Can just run BFS from $s$
▶ BFS layer $L_i = \{ \text{nodes at distance } i \text{ from } s \}$.
▶ Question: What if all the edge weights are natural numbers?

Dijkstra’s Algorithm Intuition
▶ Run BFS on augmented graph where all edge weights are the same.
▶ Let $x$ divide all edge weights $w(e)$.
▶ Split each edge into $w(e)/x$ edges of length $x$ with intermediate nodes.
▶ Keep track of layers for the nodes from the original graph.
▶ Running time? $O(n' + m')$ where $m' = \sum w(e)/x$ and $n' = n + \sum (w(e)/x - 2)$.
▶ Dijkstra’s Algorithm is a more efficient implementation of this idea.
### Dijkstra's Algorithm

- **Initialize:** Let $S = \{s\}$ be set of "explored nodes" and $d(s) = 0$.
- **While** $S \neq V$:
  - Find node $v \notin S$ that minimizes
    $$\pi(v) = \min_{(u,v) \in E, v \in S} (d(u) + w(u,v))$$
  - Add $v$ to $S$ and set $d(v) = \pi(v)$

**Running Time Analysis:** The while loop occurs $n - 1$ times and in each iteration finding $v$ can be done in $O(m)$ time. So total run time of a naive implementation is $O(mn)$ but a more clever implementation exists that uses $O(m \log n)$ time.

### Pseudocode

Let $Q = \text{Priority Queue}$, Explored = $\emptyset$.

1. Push $(s, 0)$ onto $Q$.
2. **While** $Q$ is not empty do
   - $(v, d) =$ item with smallest key from $Q$.
   - If $v$ is not marked "explored" then
     - Mark $v$ as explored and set $d[v] = d$
     - For each edge $(v, u)$ incident to $v$ do
       - Push $(u, d + w(v, u))$ onto $Q$.
   - End if
   - End if

### Proof of Correctness

- We prove by induction on $|S|$ that for all $u \in S$, $d(u)$ is the length of the shortest $s \rightarrow u$ path.
- **Base case:** When $|S| = 1$, it’s obvious since $s$ is only node in $S$ and $d(s) = 0$.
- **Inductive hypothesis:** Assume true for $|S| = k \geq 1$.
  - Let $v$ be next node added to $S$ and let $(u, v)$ be preceding edge.
  - Shortest $s \rightarrow u$ path plus $(u, v)$ is $s \rightarrow v$ path of length $\pi(v)$
  - Consider any $s \rightarrow v$ path $P$. We will show $\ell(P) \geq \pi(v)$
  - Let $(x, y)$ be the first edge in $P$ that leaves $S$, and let $P'$ be the subpath from $s$ to $x$.
  - Then,
    $$\ell(P) \geq \ell(P') + w(x, y) \geq d(x) + w(x, y) \geq \pi(y) \geq \pi(v)$$

### Minimum Spanning Tree

- Consider an undirected connected graph $G = (V, E)$ where each edge $e$ has weight $w(e)$.
- Given a subset of edges $A \subset E$, define $w(A) = \sum_{e \in A} w(e)$ to be the total weight of the edges in $A$.
- A **spanning tree** of $G$ is a tree $T$ that contains all nodes in $G$.
- **Problem:** Can we efficiently find the minimum spanning tree (MST), i.e., spanning tree with minimum total weight?
- For simplicity, we will assume all edges have distinct weights.

### Some intuition

- **Fact 1:** If all edges have unit weight, all trees are MSTs.
- **Fact 2:** Otherwise, smallest edge must be in MST.
  - Proof is an exchange argument.

### Greedy Approaches

- Consider the following greedy approaches:
  - Sort the edges by increasing weight.
    - Add next edge that doesn’t complete a cycle.
  - Sort the edges by increasing weight.
    - Let $S = \{s\}$
    - Add next edge $(u, v)$ where $u \in S, v \notin S$. Add $v$ to $S$
    - Sort the edges by decreasing weight. Remove the next edge that doesn’t disconnect the graph.
  - Which approach constructs a minimum spanning tree? All of them! We’ll prove correctness for the first two.
Important Lemma: Finding edges in MST

▷ **Cut Lemma:** Let $S \subset V$ and let $e = (u, v)$ be the lightest edge such that $u \in S$ and $v \not\in S$. The MST contains edge $e$.

▷ Note that this generalizes Fact 2 from above.

▷ Suppose $T$ is a spanning tree that doesn’t include $e$. We’ll construct a different spanning tree $T'$ such that $w(T') < w(T)$ and hence $T'$ can’t be the MST.

▷ Since $T$ is a spanning tree, there’s a $u \leadsto v$ path $P$ in $T$.

▷ Let $T' = T - \{e'\} + \{e\}$. This is a still spanning tree, since any path in $T$ that needed $e'$ can be routed via $e$ instead. But since $e$ was the lightest edge between $S$ and $V \setminus S$,

\[
w(T') = w(T) - w(e') + w(e) \leq w(T) - w(e') + w(e') = w(T)
\]

Prim’s Algorithm

▷ **Prim’s Algorithm:** Sort the edges by increasing weight.

▷ Let $S = \{s\}$.

▷ While $S \neq V$: Add next edge $(u, v)$ where $u \in S, v \not\in S$ and add $v$ to $S$.

▷ **Proof of Correctness:**

▷ Let $S$ be the set of nodes in the tree constructed so far.

▷ The next edge added to the tree is the lightest edge between $S$ and $V \setminus S$. Hence, the cut lemma implies $e$ must be in the MST.

Kruskal’s Algorithm

▷ **Kruskal’s Algorithm:** Sort the edges by increasing weight and keep on add the next edge that doesn’t complete a cycle.

▷ **Proof of Correctness:**

▷ Suppose $e = (u, v)$ is the next edge added.

▷ Let $S$ be the set of nodes that can be reached from $u$ before $e$ was added. Note that $v \not\in S$ since otherwise adding $e$ would have completed a cycle.

▷ No other edge between $S$ and $V \setminus S$ can have been encountered before since if it had it would have been added since it doesn’t complete a cycle. Hence $e$ is the lightest edge between $S$ and $V \setminus S$. Therefore, the cut lemma implies $e$ must be in the MST.